J. A. van der Pool

Optimum Storage Allocation for a File with Open Addressing

Abstract: File organizations utilizing key-to-address transformation and open addressing are studied. A simulation method and a Markov model, which were used for evaluation, are presented. The cost of retrieval as a function of storage space and accesses is also formulated. The minimum of the combined costs for different operational conditions is determined.

Introduction

In file organizations using key-to-address transformation (KAT) to locate stored records [1–3], storage is divided into physical blocks called buckets. Each bucket can contain $s \geq 1$ records of the file.

The algorithm used for KAT [3] defines a function g: $K \rightarrow A$ on the possible record keys as domain and the available bucket addresses as codomain. A record with key $k \in K$ is stored, if possible, in the bucket with address $g(k) = a$. If we load a file of n records, r_a of them will have the same image $a \in A$. If $r_a > s$, a number of records $(r_a - s)$ cannot be stored in the bucket to which they were assigned (the primary bucket).

Several methods are used to store these overflow records. In [4] and [5] studies were reported of an organization that uses a separate overflow area. In this study, the so-called open addressing method [6–8] is analyzed.

An overflow record from a bucket with address $a \in A = \{0, 1, 2, \cdots, b-1\}$, where b is number of buckets, is stored in the bucket with address $(a + d)$, where d is the smallest integer that identifies a nonfull bucket with address greater than a. In other words, when the buckets with addresses $a+1, a+2, \cdots, a+d-1$ are filled, and the bucket with address $(a + d)$ has space available, then the record is stored in $(a + d)$. If during the search for an open space, $(b - 1)$ is reached and is full, the process is continued with address 0.

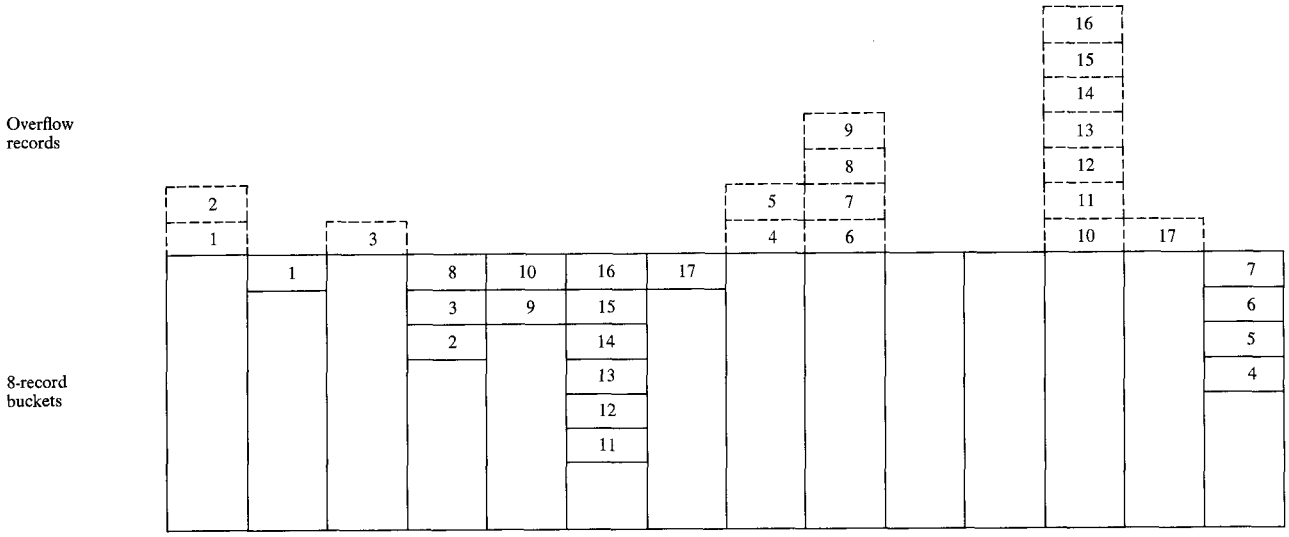
To retrieve a record, one access to the storage device is sufficient if the record is stored in its primary bucket with address $g(k) = a$. If it is not, then the buckets with addresses $a+1, a+2, \cdots, a+d$ (modulo b) must be inspected, resulting in d additional accesses. The first problem in the evaluation of open addressing is to determine the average number of additional accesses required to retrieve a record for different values of bucket size s and load factor $\ell = n/bs$. This is usually done under the assumption that the file is static, so that no additions or deletions of records occur.

Peterson [6] estimated the average number of additional accesses by detailed simulation. In this paper, we report a new simulation method. In addition, we describe how the principle of this simulation method is used to formulate a Markov model. Using both the new simulation method and the Markov model, the required additional accesses are determined and the results are compared with each other and with Peterson's simulation results.

Finally, the cost of retrieval as a function of storage space and number of accesses is formulated. Load factors giving minimal cost are then computed for different values of bucket size and different operational conditions.

Stack method for evaluating additional accesses

In an actual key-to-address transformation process, a file of n records is loaded into b buckets of size s $(n \leq bs)$. In order to calculate the average number of accesses required to retrieve stored records, we consider loading as a two phase process. In the first phase the address of each record is computed by the KAT and the records are sorted in the order of their assigned addresses. Then the records are loaded into the buckets in the order of the addresses: $0, 1, \cdots, b-1$. This special loading order and loading process facilitate the evaluation of the number of additional accesses. The values for the average obtained for the special order are however valid for any loading order, as was demonstrated by Peterson [6].



Bucket addresses a	0	1	2	3	4	5	6	7	8	9	10	11	12	13
No. of records assigned r_a	10	7	9	5	6	2	7	10	12	8	8	15	9	4
First-pass stack e_a	2	1	2	0	0	0	0	2	6	6	6	13	14	10
Second-pass stack e_a	10	10	10	9	7	1	0	0	0	0	0	0	0	0
Total	12	11	12	9	7	1	0	2	6	6	6	13	14	10

Figure 1 Records loaded in natural order and in two phases.

During the loading, a stack of overflow records is kept. Starting with $a = 0$ and the stack empty, if r_a, the number of records having the address a as result of the KAT, is greater than s, s records are loaded into the bucket and $(r_a - s)$ are added to the stack. If $s > r_a$, all records assigned to a are loaded first, and further, as many records from the stack as possible are also loaded. The number of overflow records assigned to address a is then the smaller of $(s - r_a)$ and the number of records currently in the stack. The stack of records remaining after the bucket having address $(b - 1)$ has been loaded is finally exhausted by filling the remaining open spaces in the buckets having addresses 0, 1, 2, $\cdots$, etc., in that order.

Using the concept of a stack gives us an easy way to count the total number of additional accesses required to retrieve each of the overflow records. The important property of the stack is that one need not know the identity of each record in the stack; it is sufficient for our purposes to know only the size of the stack. If there are e_a records in the stack after bucket a has been loaded, moving these records (regardless of their identity) to the next bucket contributes e_a additional accesses to the total number of overflow accesses required. The average number of additional accesses is thus

$$\bar{a} = \frac{1}{n} \sum_{a=0}^{b-1} e_a .$$

This algorithm is illustrated in Fig. 1 for the case of 112 records assigned to 14 buckets of size 8. Overflow records are numbered so that the bucket to which they were originally assigned and in which they are eventually stored can be seen. Records are stored by making two passes through the 14 buckets beginning at address $a = 0$ with the stack empty. Stack e_a is determined after each bucket has been loaded during each of the two passes. In this example after the first pass, $e_{13} = 10$. These 10 records are stored in the remaining spaces as the buckets are visited in the second pass. We can now compute on the one hand

$$\sum_{a=0}^{13} e_a = 109 .$$

On the other hand, we can count the displacements of each overflow record:

Record: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

Displacement: 1 3 1 6 6 5 5 9 10 7 8 8 8 8 8 8 8 .

The total of all displacements is again 109. With both methods we find

$$\bar{a} = \frac{109}{112} = 0.97 .$$

Simulation method

The stack provides the basis for a simulation program. We suppose that the KAT applied to a key gives a result that can be considered to be a random variable with a probability $1/b$ for each $a \in \{0, 1, \cdots, b-1\}$. Under this assumption, and for large values of b and n, we can assume that the number of records r that will be directly assigned to a bucket will have a Poisson distribution

$$P(r) = e^{-m}\frac{m^r}{r!}$$

with $m = n/b$, the average number of records assigned to a bucket. The simulation can now be done by drawing successive random numbers r_a for each bucket ($a = 0, 1, \cdots, b-1$) such that these numbers have a Poisson distribution (see, for example, [9]). After each drawing of r_a, the resulting stack e_a can be calculated. If $e_{b-1} > 0$, then the stack is "carried around" to address 0. The previously calculated e_0, e_1, etc., are corrected until e_{b-1} is exhausted by filling remaing open spaces.

Markov model

The deletions from and additions to the stack can be considered as trials in a Markov chain; see, for example, [10]. The number of records in the stack determines the state of the system considered. For bucket size s ($s \geq 1$), the (infinite) matrix of transition probabilities is:

$$\mathbf{M} = \begin{Bmatrix} \sum_{i=0}^{s} P(i) & P(s+1) & P(s+2)\cdots \\ \sum_{i=0}^{s-1} P(i) & P(s) & P(s+1)\cdots \\ \vdots & \vdots & \vdots \\ \sum_{i=0}^{1} P(i) & P(2) & P(3)\cdots \\ P(0) & P(1) & P(2)\cdots \\ 0 & P(0) & P(1)\cdots \\ 0 & 0 & P(0)\cdots \\ \vdots & \vdots & \vdots \end{Bmatrix}.$$

Let $\mathbf{u} = \{u_0, u_1, \cdots\}$ be the steady-state solution of

$$\mathbf{uM} = \mathbf{u}. \tag{1}$$

Then the average stack length is

$$\bar{e} = \sum_{i=0}^{\infty} i u_i. \tag{2}$$

The average number of additional accesses is

$$\bar{a} = \frac{b\bar{e}}{n} = \frac{\bar{e}}{m}. \tag{3}$$

Schay and Spruce [8] found the same matrix for the case $s = 1$, based on a somewhat different loading method. They then derived an analytic solution for $\mathbf{u}$ and $\bar{a}$. Konheim and Weiss [7] also deal with the case $s = 1$ and find $E(s_k)$, the expectation for the number of additional accesses for the kth record. Then they find

$$\bar{a} = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} E(s_k),$$

keeping the ratio $\ell = n/b$ constant. The result found in both papers is

$$\bar{a} = \frac{\ell}{2(1-\ell)}. \tag{4}$$

Tainter [11] extended the work of Schay and Spruce to arbitrary s. Using generating functions he derives s linear equations in s unknowns to obtain $u_0, i_1, \cdots, u_{s-1}$ and a recursive relation to find the other components of the steady state vector. Also, the author thanks Dr. A. Konheim of the IBM T. J. Watson Research Center for pointing out to him that the same matrix of transition probabilities occurs in a queuing problem arising in [12]. The detailed analysis given in Konheim's paper with Meister is therefore also applicable for finding the stationary distribution for the stack.

In this paper a direct numerical approach to finding $\mathbf{u}$ and $\bar{a}$ is used.

Minimum of the cost function

The cost of retrieval from the file can be formulated as follows. We define

c_s = cost of storing one record during a unit of time,
α = file activity (the fraction of records accessed during the unit of time),
c_p = cost of a regular access,
c_a = cost of an additional access.

Then the cost of storage and retrieval is

$$C(b,s;\alpha) = bsc_s + \alpha nc_p + \alpha n\bar{a}c_a. \tag{5}$$

It is assumed that c_p is constant. Furthermore, the variable part of C can be normalized by dividing it by nc_s, the cost for storing n records. The resulting relative variable cost function is

$$\frac{1}{\ell} + \gamma \cdot \bar{a}, \tag{6}$$

where

$$\gamma = \frac{\alpha c_a}{c_s}.$$

It is convenient to factor out the variable γ since it represents a ratio of application-dependent variables. This application factor is discussed by the author in [5].

For the case $s = 1$, we can substitute (4) into (6) and find the condition for a minimum:

$$-\frac{1}{\ell^2}+\frac{\gamma}{2}\cdot\frac{1}{(1-\ell)^2}=0,$$

which has the solution

$$\ell=\frac{1}{1+\sqrt{\gamma/2}}. \tag{7}$$

Thus, for $s = 1$ the minimum of the relative cost function is

$$1+\sqrt{2\gamma} \tag{8}$$

The numerical method for computing the optimum value of ℓ and the corresponding value of the relative cost function with $s > 1$ is explained later.

Computations

Computations were made using APL and an IBM System 370 Model 195. The computations were made for bucket sizes of $s = 1, 2, 3, 5, 10, 20, 40$ and for different load factors ℓ. The simulations were performed first. For a given s and ℓ the number of buckets was selected to be $b = [100{,}000/m]$ with $m = \ell s$. Then a sequence of b drawings from a Poisson distribution with parameter m was made. After each sequence, the relative frequency function of the numbers in the stack was obtained.

The second relative frequency function was used to compute $\bar{a}$. For each case 10 sequences were used. At the end of ten sequences the overall average and the standard deviation in the ten results for $\bar{a}$ were obtained.

A second program was written to compute the Markov-derived steady-state vector $\mathbf{u}$ from $\mathbf{uM} = \mathbf{u}$. The method of iterated vectors was used. For a certain s the vectors were computed for increasing values of ℓ. For $\ell = 0.5$, the begin vector chosen was $\mathbf{u}^{(1)} = \{1, 0, 0, \cdots\}$. For higher values of ℓ, the final result of the preceding case was taken as begin vector. Then, the successive iterations

$$\mathbf{u}^{(n+1)} = \mathbf{u}^{(n)}\mathbf{M}; \qquad n = 1, 2, \cdots$$

were performed.

The terms of $\mathbf{M}$ are obtained from a table of values of $P(r;m)$. The terms of this table were computed up to and including the first one less than $0.5 \cdot 10^{-6}$. After each vector matrix multiplication, the sum of the elements of $\mathbf{u}^{(n+1)}$ was normalized to one. The iterations were performed until the maximum absolute difference between elements of $\mathbf{u}^{(n+1)}$ and $\mathbf{u}^{(n)}$ was less than $0.5 \cdot 10^{-6}$. With the final resulting $\mathbf{u}$ we computed the average stack length $\bar{e}$ and the average number of additional accesses $\bar{a}$. Further, the standard deviation of the stack length,

$$\sigma_e=\sqrt{\sum_{i=0}^{\infty} u_i i^2 - e^{-2}},$$

was computed.

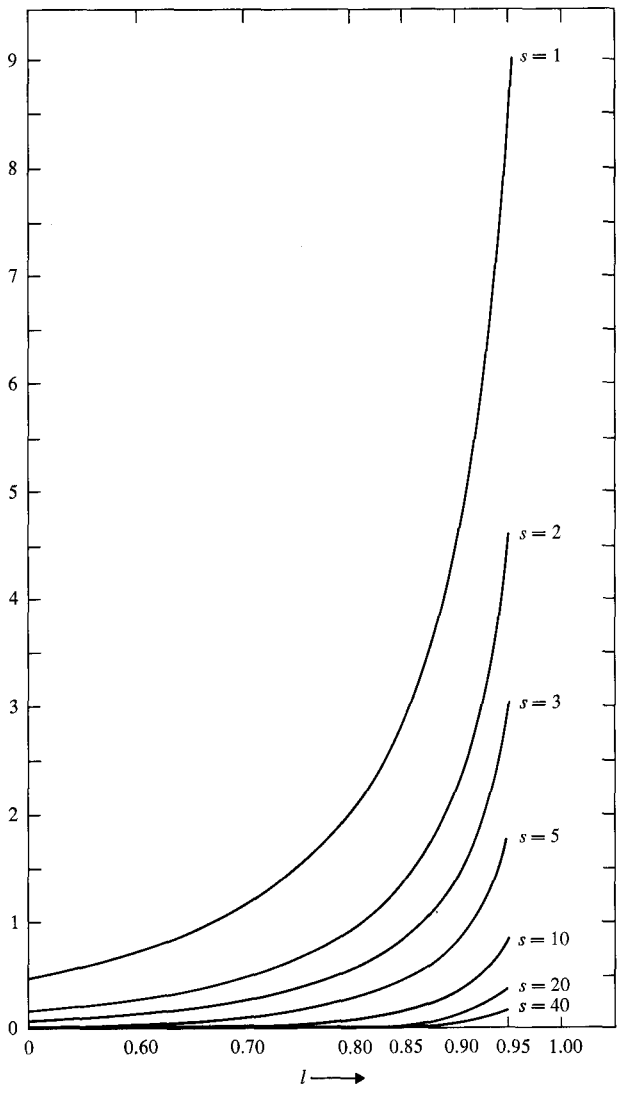


Figure 2 Number of additional accesses in open addressing.

The computations using the Markov model are in most cases much faster than the simulation. The computation time for all cases listed in Table 1 was 2 minutes and 34 seconds on the Model 195. For values of ℓ very close to one, the number of iterations becomes very large, however.

To compute $\mathbf{u}$, parabolas were fitted through successive groups of three points of the cost function with abscissas $m_1 < m_2 < m_3$. The abscissa m_2 of each following group was the abscissa of the minimum of the parabola determined by the previous group. Abscissas m_1 and m_3 were chosen so that the length of the interval

Table 1 Additional accesses.

Bucket size s	Load factor	m	Markov model			Simulation		Peterson Add. accesses average
			Stack length		Add. accesses average	Add. accesses		
			Average	St. dev.		Average	St. dev.	
1	0.50	0.50	0.250	0.629	0.500			0.541
	0.60	0.60	0.450	0.912	0.750			0.832
	0.70	0.70	0.817	1.365	1.167	1.150	0.019	1.260
	0.80	0.80	1.600	2.238	2.000			2.223
	0.85	0.85	2.407	3.092	2.832			
	0.90	0.90	4.046	4.775	4.495	4.465	0.207	4.526
	0.91	0.91	4.595	5.331	5.049			
	0.92	0.92	5.281	6.024	5.740			
	0.93	0.93	6.163	6.909	6.626			
	0.94	0.94	7.336	8.081	7.804			
	0.95	0.95	8.972	9.703	9.444	9.349	1.145	
2	0.50	1.00	0.177	0.550	0.177			
	0.60	1.20	0.352	0.838	0.293			0.325
	0.70	1.40	0.691	1.301	0.494	0.495	0.010	0.517
	0.80	1.60	1.445	2.189	0.903			0.927
	0.85	1.70	2.238	3.053	1.316			
	0.90	1.80	3.863	4.751	2.146	2.150	0.149	2.148
	0.91	1.82	4.409	5.312	2.423			
	0.92	1.84	5.094	6.010	2.768			
	0.93	1.86	5.976	6.905	3.213			
	0.94	1.88	7.152	8.091	3.804			
	0.95	1.90	8.798	9.740	4.631	4.764	0.510	4.112
3	0.50	1.50	0.131	0.486	0.087			
	0.60	1.80	0.285	0.776	0.158			
	0.70	2.10	0.601	1.245	0.286	0.285	0.006	
	0.80	2.40	1.329	2.145	0.554			
	0.85	2.55	2.108	3.017	0.827			
	0.90	2.70	3.719	4.726	1.377	1.362	0.061	
	0.91	2.73	4.263	5.290	1.562			
	0.92	2.76	4.945	5.992	1.792			
	0.93	2.79	5.825	6.891	2.088			
	0.94	2.82	7.001	8.084	2.483			
	0.95	2.85	8.649	9.744	3.035	3.012	0.192	
5	0.50	2.50	0.077	0.387	0.031			
	0.60	3.00	0.198	0.673	0.066			0.072
	0.70	3.50	0.474	1.149	0.136	0.136	0.002	0.131
	0.80	4.00	1.156	2.066	0.289			0.280
	0.85	4.25	1.910	2.952	0.449			0.443
	0.90	4.50	3.494	4.679	0.777	0.771	0.045	0.762
	0.91	4.55	4.033	5.247	0.886			
	0.92	4.60	4.710	5.954	1.024			
	0.93	4.65	5.585	6.859	1.201			
	0.94	4.70	6.757	8.060	1.438			
	0.95	4.75	8.403	9.733	1.769	1.649	0.156	1.467
10	0.50	5.00	0.024	0.228	0.005			
	0.60	6.00	0.092	0.488	0.015			0.016
	0.70	7.00	0.293	0.959	0.042	0.042	0.001	0.042
	0.80	8.00	0.879	1.900	0.110			0.111
	0.85	8.50	1.577	2.810	0.185			0.172
	0.90	9.00	3.101	4.572	0.345	0.341	0.011	0.330
	0.91	9.10	3.627	5.149	0.399			
	0.92	9.20	4.292	5.865	0.467			
	0.93	9.30	5.154	6.780	0.554			
	0.94	9.40	6.314	7.993	0.672			
	0.95	9.50	7.949	9.681	0.837	0.843	0.069	0.755

Table 1 Additional accesses. (Continued)

Bucket size s	Load factor	m	Markov model: Stack length: Average	Markov model: Stack length: St. dev.	Markov model: Add. accesses average	Simulation: Add. accesses: Average	Simulation: Add. accesses: St. dev.	Peterson Add. accesses average
20	0.50	10.00	0.003	0.083	0.000			
	0.60	12.00	0.025	0.268	0.002			0.002
	0.70	14.00	0.134	0.693	0.010	0.010	0.001	0.010
	0.80	16.00	0.574	1.636	0.036			0.033
	0.85	17.00	1.179	2.573	0.069			0.066
	0.90	18.00	2.597	4.383	0.144	0.144	0.011	0.134
	0.91	18.20	3.100	4.973	0.170			
	0.92	18.40	3.742	5.703	0.203			
	0.93	18.60	4.581	6.634	0.246			
	0.94	18.80	5.717	7.864	0.304			
	0.95	19.00	7.327	9.572	0.386	0.400	0.042	0.334
40	0.50	20.00	0.000	0.011	0.000			
	0.60	24.00	0.002	0.084	0.000			
	0.70	28.00	0.035	0.377	0.001	0.001	0.000	
	0.80	32.00	0.290	1.248	0.009			
	0.85	34.00	0.752	2.196	0.022			
	0.90	36.00	1.990	4.062	0.055	0.056	0.006	
	0.91	36.40	2.453	4.669	0.067			
	0.92	36.80	3.053	5.420	0.083			
	0.93	37.20	3.849	6.374	0.103			
	0.94	37.60	4.940	7.631	0.131			
	0.95	38.00	6.506	9.370	0.171	0.167	0.018	

(m_1,m_3) decreased. The process was terminated when $|m_3 - m_1| < 0.0005$.

Analysis of results

The results obtained by the stack-oriented simulation and with the Markov model are listed in Table 1. For comparison, the results of Peterson [6] are also listed. The values of $\bar{a}$ computed with the Markov model are plotted in Fig. 2. No effort was made to investigate the sample distribution of the average number of additional accesses. However, to the extent that the variance computed from the simulation runs is a good estimate, the results of the Markov model and the simulation are in good agreement. The same may be said in relation to Peterson's results, taking into consideration the much smaller number of records used in his simulations.

As we see, the standard deviation of the stack length is rather large. The relative frequency functions of the stack size are J-shaped with the maximum frequency at zero, and a very steep drop from zero to one. When the load factor is increasing, the "tail" becomes more extended and very long when ℓ approaches 1. In Figs. 3 and 4 some examples are given for $s = 1$ and $s = 10$, respectively.

For $s = 1$, the values computed by the method of iterated vectors can be compared with the results from Eq. (4):

ℓ	$\frac{\ell}{2(1-\ell)}$	Iterated vectors
0.50	0.500	0.500
0.60	0.750	0.750
0.70	1.167	1.167
0.80	2.000	2.000
0.85	2.833	2.832
0.90	4.500	4.495
0.91	5.055	5.049
0.92	5.750	5.740
0.93	6.643	6.626
0.94	7.833	7.804
0.95	9.500	9.444

For values of ℓ close to 1, the method of iterated vectors gives results that are too low; but up to $\ell = 0.9$, the method is certainly adequate for the purposes of this study.

In Table 2, the minimum values are listed. For comparison, the corresponding values for an independent overflow area [4] are given.

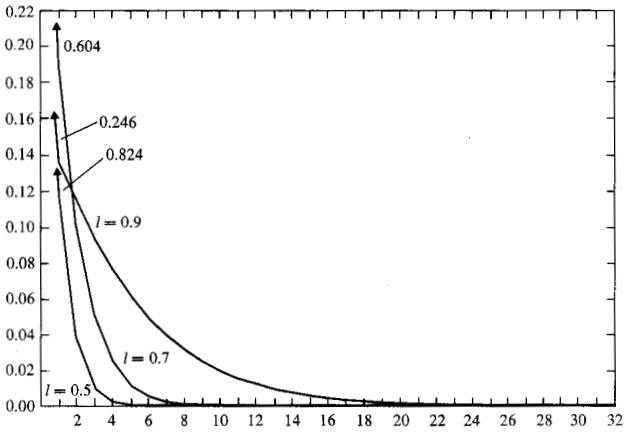
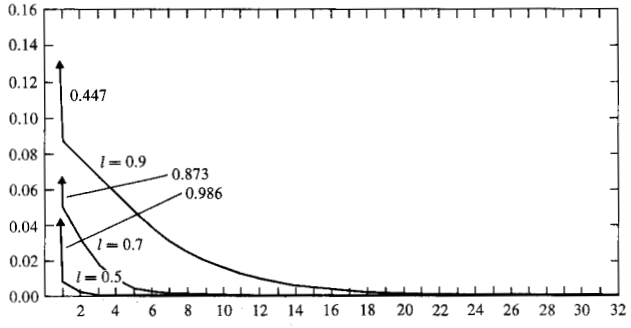


Figure 3 Relative frequency functions for stack size when bucket size is $s = 1$.

Figure 4 Relative frequency functions for stack size when bucket size is $s = 10$.



For $s = 1$ we can again compare the results of the iterated vectors with the results of formulas (7) and (8). For all values of γ except $\gamma = 0.01$, there is agreement to three decimal places. For $\gamma = 0.01$, the values of the minimum cost agree, but (7) and (8) give 0.934 and 7.875 for ℓ and $\bar{a}$, respectively.

We see that for the same value of γ the cost in the case of open addressing is always larger than when an overflow area is used. However, a comparison for equal values of γ does not give a proper evaluation of the two methods. In case of an overflow area with chained overflow records, the cost of an additional access will be, in general, higher than for open addressing. In the first case, each access will require a search and a read command, often preceded by a seek command [14]. In the second case, the seek will be avoided most of the time and the bucket for inspection will be found on the same track. If central storage space allows, a whole track may be read on a primary access. The additional access to several additional buckets is then performed in main storage.

Let γ_1 and γ_2 be the application factors for the overflow area method and open addressing, respectively. In a specific case, $\gamma_2 = \rho \cdot \gamma_1$ with $\rho < 1$. The critical value ρ_c of ρ is the value with the property that the open addressing and the other method have the same minimum cost.

The γ giving minimum cost R for a given certain bucket size s can be computed with the formulas derived in [5]. By solving for γ in the expression for the relative cost function [5] we find for cost R

$$\gamma = \frac{R - \dfrac{s + \bar{l}}{m}}{\bar{a}}.$$

We note that the right-hand side is for given R a function $\phi(\gamma;R)$ of γ only; (m is determined by the minimum cost condition).

By using

$$\gamma_{n+1} = \phi(\gamma_n;R)$$

as iteration procedure, a solution for γ is easily found.

Pairs of application factors giving the same minimum cost are listed in Table 3. From this table we see that in general ρ_c becomes smaller when

1. Application factor γ becomes smaller
2. Bucket size becomes larger.

When designing an overflow handling technique, a second consideration of importance is the use of pointers. When a separate overflow area is used, an address pointer is required from the primary bucket to the beginning of the overflow chain, and, further, from record to record in the chain. In the case of open addressing, no pointers are required. The pointers require storage space that was not taken into consideration in [4]. The relative cost of the required pointers is a function of bucket size and record length. The relative overhead tends to be high when the bucket size is small and when the records are short.

The choice between the two methods is application dependent. The conditions that tend to favor open addressing are

1. Large value of application factor γ
2. Short record length
3. Circumstances that prescribe a small bucket size.

The two last reasons will, in general, not go together, because a short record length will usually make a larger bucket size desirable.

An interesting area for further investigation is the development of hybrid open addressing/chained overflow techniques. Overflow records for example might be

Table 2 Minimum cost load factors.

Bucket size s	γ	Open addressing				Overflow area				
		m	Load factor	Add. acc.	Min. cost	m	Load factor	Over-flow	Add. acc.	Min. cost
1	2.00	0.500	0.500	0.500	3.000	0.883	0.883	0.336	0.441	2.351
	1.00	0.586	0.586	0.707	2.414	1.163	1.163	0.409	0.581	1.850
	0.50	0.667	0.667	1.000	2.000	1.496	1.496	0.481	0.748	1.524
	0.10	0.817	0.817	2.237	1.447	2.445	2.445	0.626	1.222	1.158
	0.05	0.863	0.863	3.160	1.316	2.914	2.914	0.675	1.457	1.091
	0.01	0.932	0.932	6.864	1.141	4.104	4.104	0.760	2.052	1.025
2	2.00	1.023	0.511	0.375	2.705	1.586	0.793	0.202	0.294	2.052
	1.00	1.185	0.593	0.565	2.252	1.977	0.988	0.267	0.424	1.703
	0.50	1.341	0.670	0.842	1.913	2.428	1.214	0.337	0.589	1.456
	0.10	1.636	0.818	2.048	1.428	3.664	1.832	0.494	1.098	1.149
	0.05	1.727	0.863	2.958	1.306	4.255	2.127	0.551	1.359	1.089
	0.01	1.868	0.934	6.861	1.139	5.705	2.853	0.654	2.027	1.025
3	2.00	1.570	0.523	0.302	2.514	2.317	0.772	0.144	0.227	1.893
	1.00	1.800	0.600	0.475	2.142	2.795	0.932	0.200	0.344	1.617
	0.50	2.024	0.675	0.736	1.851	3.339	1.113	0.264	0.501	1.412
	0.10	2.459	0.820	1.928	1.413	4.797	1.599	0.416	1.011	1.143
	0.05	2.593	0.864	2.831	1.298	5.481	1.827	0.475	1.282	1.086
	0.01	2.801	0.934	6.663	1.138	7.134	2.378	0.584	1.987	1.025
5	2.00	2.725	0.545	0.219	2.273	3.834	0.767	0.092	0.162	1.719
	1.00	3.071	0.614	0.366	1.994	4.454	0.891	0.134	0.260	1.517
	0.50	3.414	0.683	0.599	1.764	5.147	1.029	0.187	0.400	1.358
	0.10	4.103	0.821	1.719	1.391	6.963	1.393	0.325	0.893	1.132
	0.05	4.325	0.865	2.607	1.286	7.799	1.560	0.382	1.170	1.082
	0.01	4.671	0.934	6.453	1.135	9.778	1.956	0.494	1.910	1.024
10	2.00	5.868	0.587	0.133	1.970	7.813	0.781	0.048	0.102	1.531
	1.00	6.433	0.643	0.240	1.795	8.697	0.870	0.075	0.175	1.400
	0.50	7.012	0.701	0.424	1.638	9.669	0.967	0.112	0.287	1.289
	0.10	8.246	0.825	1.412	1.354	12.158	1.216	0.221	0.734	1.117
	0.05	8.667	0.867	2.248	1.266	13.278	1.328	0.271	1.005	1.074
	0.01	9.345	0.935	6.033	1.130	15.876	1.588	0.376	1.768	1.023
20	2.00	12.787	0.639	0.078	1.720	16.156	0.808	0.025	0.064	1.391
	1.00	13.652	0.683	0.149	1.614	17.417	0.871	0.041	0.117	1.306
	0.50	14.565	0.728	0.281	1.514	18.787	0.939	0.064	0.203	1.230
	0.10	16.641	0.832	1.090	1.311	22.245	1.112	0.143	0.588	1.101
	0.05	17.408	0.870	1.844	1.241	23.781	1.189	0.183	0.841	1.066
	0.01	18.697	0.935	5.442	1.124	27.285	1.364	0.273	1.598	1.022
40	2.00	27.857	0.696	0.046	1.527	33.518	0.838	0.012	0.041	1.288
	1.00	29.124	0.728	0.091	1.464	35.308	0.883	0.022	0.078	1.233
	0.50	30.481	0.762	0.179	1.402	37.239	0.931	0.035	0.142	1.181
	0.10	33.754	0.844	0.793	1.264	42.086	1.052	0.089	0.460	1.085
	0.05	35.065	0.877	1.426	1.212	44.224	1.106	0.119	0.688	1.057
	0.01	37.435	0.936	4.749	1.116	49.054	1.226	0.190	1.416	1.020

stored in adjacent buckets (in the open addressing way), but not farther from the primary bucket than d_{max}. Records still in the overflow stack when this limit is reached are stored by chaining in an overflow area.

This may bring about the following advantages:

1. Up to a certain value of d_{max}, the value of γ_2 may be considerably lower than γ_1.
2. The influence of the long tails in the relative frequency function of the stack on the number of additional accesses is reduced.

Alternatively open buckets might be chained in an overflow area.

Table 3 Equivalent pairs of application factors.

Bucket size s	Application factor for: Open addressing γ_2	Application factor for: Overflow area γ_1	Critical factor $\rho_c = \gamma_2/\gamma_1$	Load factor: Open addressing	Load factor: Overflow area	Minimum cost
1	2.00	3.691	0.542	0.500	0.679	3.000
	1.00	2.144	0.466	0.586	0.857	2.414
	0.50	1.270	0.394	0.667	1.060	2.000
	0.10	0.401	0.249	0.817	1.612	1.447
	0.05	0.250	0.200	0.863	1.887	1.316
	0.01	0.087	0.115	0.932	2.540	1.141
5	2.00	8.276	0.242	0.545	0.561	2.273
	1.00	4.297	0.233	0.614	0.648	1.994
	0.50	2.290	0.218	0.683	0.744	1.764
	0.10	0.587	0.170	0.821	0.996	1.391
	0.05	0.338	0.148	0.865	1.113	1.286
	0.01	0.103	0.097	0.934	1.385	1.135
20	2.00	20.546	0.097	0.639	0.638	1.720
	1.00	10.026	0.100	0.683	0.684	1.614
	0.50	4.974	0.101	0.728	0.734	1.514
	0.10	1.044	0.096	0.832	0.867	1.311
	0.05	0.556	0.090	0.870	0.929	1.241
	0.01	0.144	0.069	0.935	1.072	1.124

References

1. W. Buchholz, "File Organization and Addressing," *IBM Systems Journal* **2,** 86–111 (June 1963).
2. R. Morris, "Scatter Storage Techniques." *Communications of the ACM* **11,** No. 1, 38–44 (January 1968).
3. V. Y. Lum, P. S. T. Yuen, and M. Dodd, "Key-to-Address Transform Techniques: A Fundamental Performance Study on Large Existing Formatted Files," *Communications of the ACM* **14,** No. 4, 228–239 (April 1971).
4. J. A. van der Pool, "Optimal Storage Allocation for Initial Loading of a File," *IBM J. Res. Develop.* **16,** No. 6, 579 (1972).
5. J. A. van der Pool, "Optimal Storage Allocation for a File in Steady State," *IBM J. Res. Develop.* **17,** No. 1, 27 (1973).
6. W. W. Peterson, "Addressing for Random-Access Storage," *IBM J. Res. Develop.* **1,** No. 2, 130–146 (April 1957).
7. A. G. Konheim and B. Weiss, "An Occupancy Discipline and Applications," *SIAM J. Appl. Math.* **14,** No. 6, 1266–1272 (November 1966).
8. G. Schay and W. G. Spruce, "Analysis of a File Addressing Method," *Communications of the ACM* **5,** No. 8, 459–462 (August 1962).
9. D. E. Knuth, *The Art of Computer Programming*, Vol. 2 "Seminumerical Algorithms," Addison-Wesley Publishing Co., 1969, 117–118.
10. W. Feller, *An Introduction to Probability Theory and Its Applications*, John Wiley & Sons, Inc., New York, 338 ff.
13. E. Bodewig, *Matrix Calculus*, North Holland Publishing Co., Amsterdam, 296 ff.
14. *Introduction to IBM System/360 Direct Access Storage Devices and Organization Methods*, Form No. GC20-1649, IBM Corporation, Data Processing Division, White Plains, New York.
11. M. Tainiter, "Addressing for Random Access Storage with Multiple Bucket Capacities," *Journal of the ACM* **10,** 307–315 (1963).
12. A. G. Konheim and B. Meister, "Service in a Loop System," *Journal of the ACM* **19,** 92–108 (1972).

Received March 9, 1972; revised January 9, 1973

The author is located at the head office of IBM Nederland N.V. Amsterdam, The Netherlands.